2024

Rapport final du projet de développement

KINANE-DAOUADJI ISAAC NISTOR IARINA KHAMIS YARA ZHOU JEREMY
BENHADDOU CHADY

CHARGÉ DE TD : M. NICHOLAS MAUDET

Remerciement

Nous tenons à remercier nos clients et plus particulièrement M Maudet pour ses explications et le temps qu'il a bien voulu nous consacré. Nous remercions aussi M Thieyre pour ses conseils.

Ce rapport vise à fournir une vue d'ensemble complète du projet, en mettant l'accent sur ses principales caractéristiques, son architecture, son processus de développement, ses tests et son impact environnemental. Créé pour être concis et informatif, ce rapport complète la documentation technique en fournissant des informations supplémentaires sur l'utilisation du logiciel, la justification des choix de développement, les tests effectués et l'évaluation de l'empreinte carbone du projet.

Structure du Rapport :

Le manuel utilisateur :

Ce manuel est conçu pour être clair et facile à suivre. Nous avons organisé les informations de manière logique et structurée afin de faciliter la navigation et la compréhension. Vous y retrouverez des explications détaillées sur les fonctionnalités du logiciel ainsi que des illustrations pour guider les utilisateurs à travers les différentes étapes.

Cette partie contient plusieurs chapitres :

1. Chapitre 1 : Introduction au Logiciel
2. Chapitre 2 : Installation et Configuration
3. Chapitre 3 : Interface Utilisateur et Fonctionnalités Principales

Chapitre 1 : Introduction au Logiciel :

Dans ce chapitre, nous vous présenterons une vision globale du logiciel, incluant ses objectifs et ses avantages. Ce logiciel utilise diverses méthodes de vote pour déterminer les

vainqueurs. Il a été développé dans le but de fournir une solution performante et précise pour déterminer les vainqueurs dans différents contextes électoraux.

L'objectif principal de notre logiciel est de simplifier le processus de détermination des vainqueurs en utilisant différentes méthodes de vote telles que la pluralité, borda, le scrutin à vote unique transférable (STV), le vote par approbation, etc. Notre logiciel offre une interface accueillante et intuitive qui permet aux utilisateurs de saisir facilement les données de vote et d'obtenir rapidement les résultats.

Les avantages de notre logiciel incluent sa précision dans le calcul des résultats, sa flexibilité pour prendre en charge différentes méthodes de vote, ainsi que sa facilité d'utilisation pour les utilisateurs de tous niveaux de compétence.

Chapitre 2 : Installation et Configuration :

Ce chapitre vous guidera à travers les étapes nécessaires pour une installation réussie.

Pour utiliser notre logiciel, il est nécessaire d'installer PySide6. Vous pouvez suivre ces étapes simples pour effectuer l'installation :
Tout d'abord, assurez-vous d'avoir Python installé sur votre système. Ensuite, ouvrez votre terminal ou votre invite de commande et utilisez pip, le gestionnaire de paquets Python, pour installer PySide6 en exécutant la commande suivante : 'pip install PySide6'. Cette commande téléchargera et installera automatiquement PySide6 ainsi que ses dépendances nécessaires. Une fois l'installation terminée, vous pouvez vérifier si PySide6 a été correctement installé en exécutant la commande 'pyside6 –version', qui devrait afficher la version de PySide6 installée sur votre système. Avec PySide6 installé, vous êtes prêt à utiliser notre logiciel et à profiter de ses fonctionnalités.

Chapitre 3 : Interface Utilisateur :

Ce chapitre vous guidera dans la découverte de l'interface, vous permettant de vous familiariser avec ses différents éléments, ses menus et ses fonctionnalités et de naviguer aisément dans le logiciel.

Pour lancer le logiciel entrez dans le terminal la commande './run_app.sh'.

Dans la fenêtre principale, plusieurs options sont disponibles pour configurer la Map.
Vous avez la possibilité d'ajouter une population en spécifiant le rayon et la position du centre du cercle (x, y) et en appuyant sur 'Soumettre', ou en cliquant directement sur la grille et en entrant le nombre d'individus composant la population avec un rayon prédéfini.
Pour ajuster le rayon du cercle, vous pouvez cliquer sur 'Ajouter 0.1' pour l'augmenter ou sur 'Ajouter -0.1' pour le diminuer et, pour supprimer la population, cliquez simplement sur 'Supprimer'. (Figure 1)
Vous avez également le choix du type de génération que vous souhaitez : uniforme, beta, exponentielle ou triangulaire (il faut sélectionner le type avant de choisir la zone sur la map).

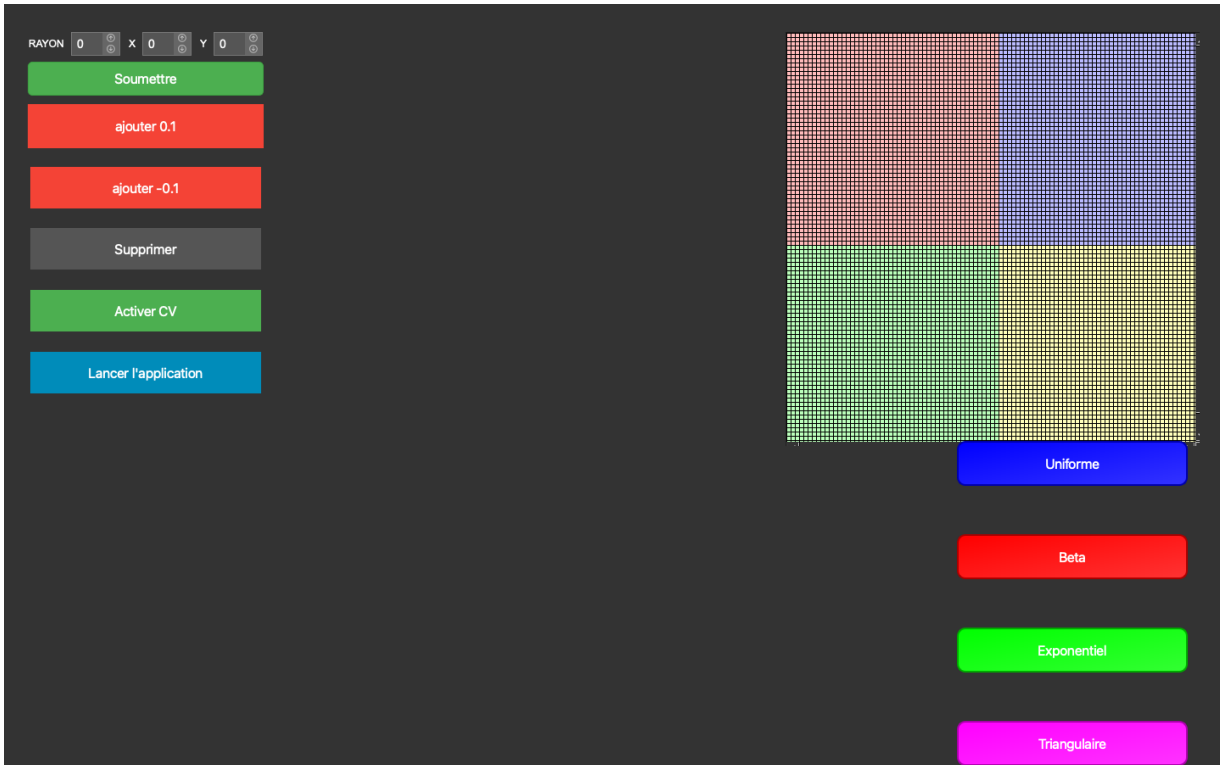


Figure 1 : Fenêtre principale.

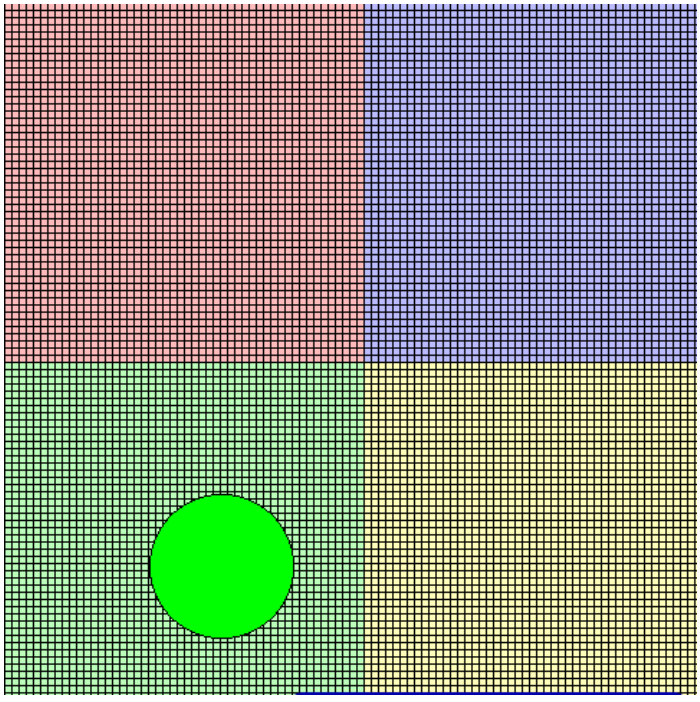


Figure 2 : Map après l'ajout de la population en la générant exponentiellement.

Le bouton 'Activer CV' qui se situe au-dessous du bouton 'Supprimer' (Figure 1), permet d'activer le système de vision par ordinateur qui permet d'augmenter et de diminuer avec les doigts la taille du cercle

Figure 3 : Passage au mode augmentation

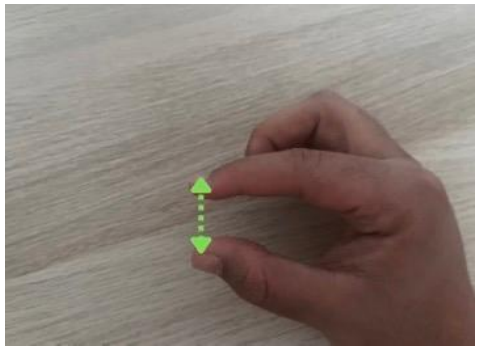


Figure 4 : Augmentation légère

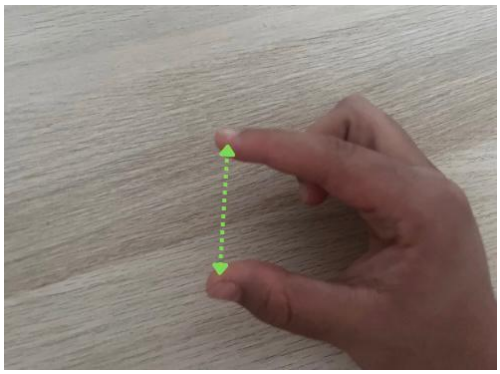


Figure 5 : Augmentation rapide

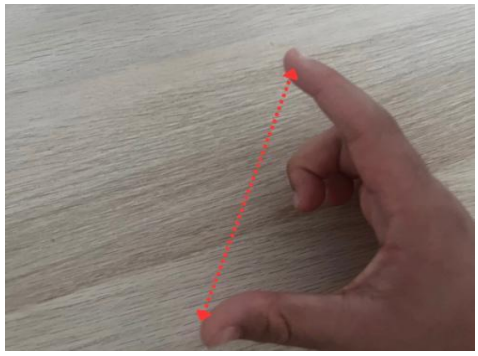


Figure 6 : Passage au mode diminution

Une fois fini et que tous les paramètres ont été choisis il suffit de cliquer sur 'Lancer l'application' et d'ici il faut choisir entre une map 2D ou 3D.

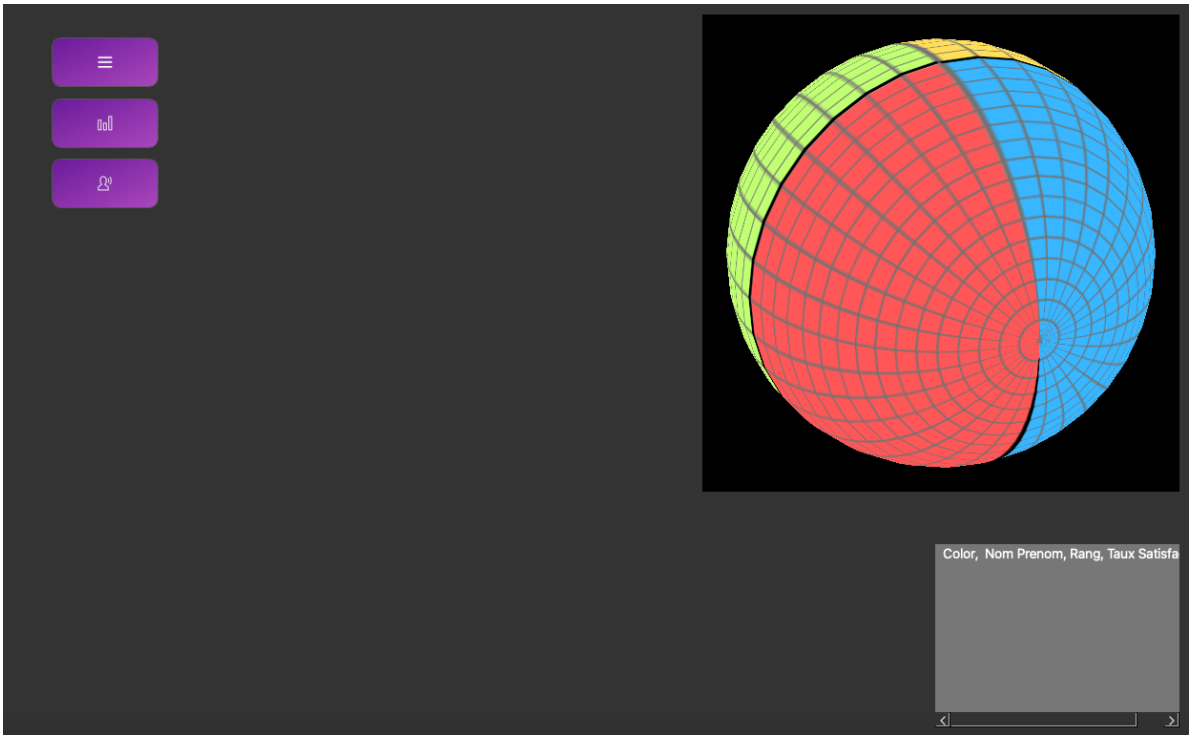


Figure 7 : Interface utilisateur.

Figure 8 : Bouton Menu

Après avoir cliqué sur le bouton Menu, ces boutons seront affichés.

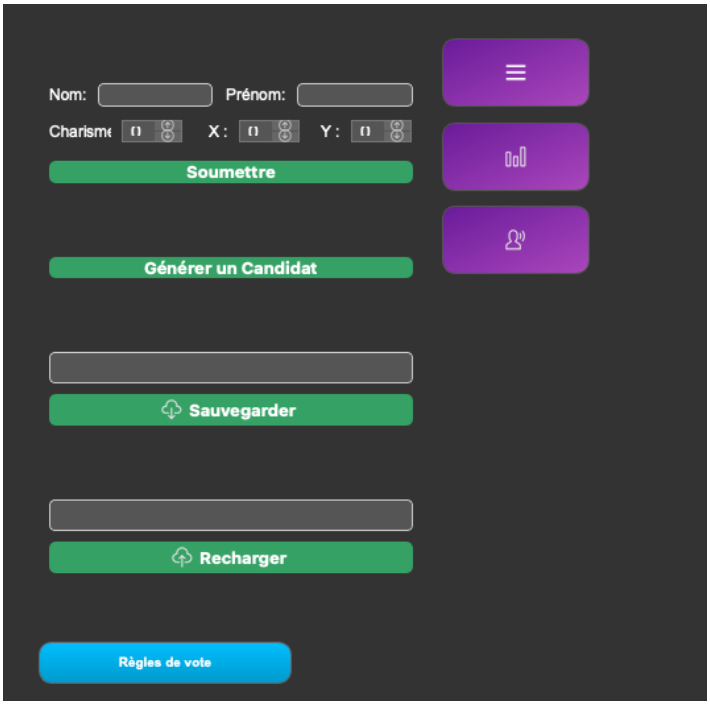


Figure 9 : Boutons pour des différentes fonctionnalités.

Dorénavant, vous avez la possibilité de générer un candidat soit de manière aléatoire en appuyant sur 'Générer un candidat' (le nom, le prénom, le charisme et les coordonnées seront générer aléatoirement), soit manuellement ; vous pourrez ainsi personnaliser chaque détail selon vos préférences en les saisissant directement dans les champs dédies.

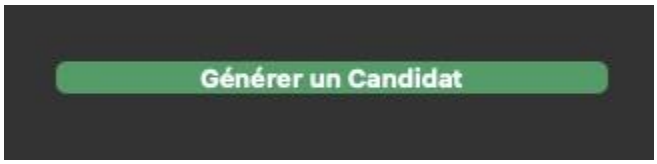


Figure 10 : Bouton pour générer un candidat aléatoirement.

Les candidats générés apparaîtront sur la Map en fonction de leurs coordonnées. De plus, leurs caractéristiques seront affichées dans le tableau situé en dessous de la carte (Figure 7), permettant de visualiser leur couleur respective pour les distinguer, ainsi que leur nom, prénom, rangs, et enfin leur taux de satisfaction.

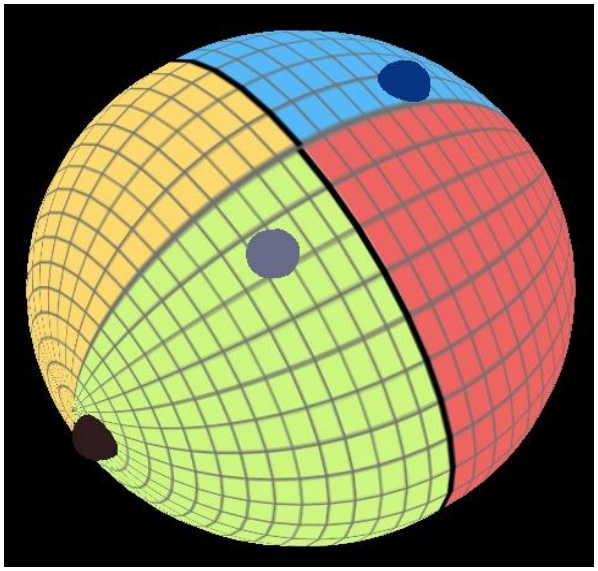


Figure 11 : Les candidats générés sur la Map 3D.

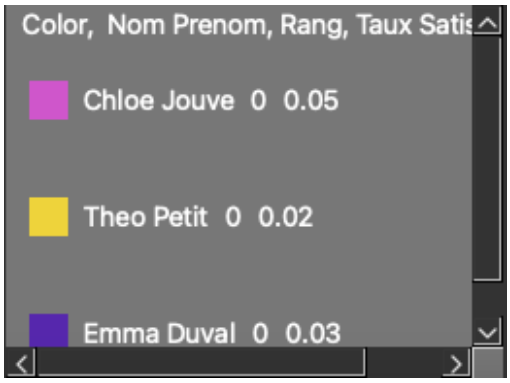


Figure 12 : Tableau des candidats et leurs caractéristiques.

Ensuite, en appuyant sur le bouton "Règle de vote", les différentes méthodes de votes s'afficheront, telles que Copeland, Borda, Pluralité, STV et Approbation.

Pour déterminer un vainqueur, il est nécessaire de choisir l'une des méthodes de vote proposées ; le vainqueur sera sélectionné selon cette méthode.
Un trophée apparaîtra à côté du nom du vainqueur dans le tableau des candidats, le désignant ainsi de manière distincte.

Figure 13 : Les différentes règles de vote.

Figure 14 : Vainqueur selon la méthode Pluralité

Pour sauvegarder les candidats et les individus générés, veuillez insérer le nom du fichier juste au-dessus du bouton "Sauvegarder" (Figure 4), puis appuyez sur ce bouton pour finaliser la sauvegarde.
Pour recharger un fichier déjà sauvegardé, veuillez insérer son nom juste au-dessus du bouton "Recharger", puis cliquez sur ce bouton pour effectuer le chargement.

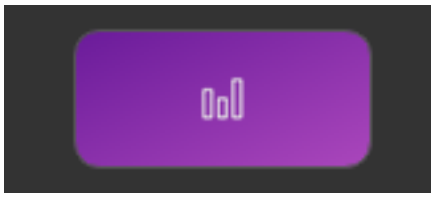


Figure 15 : Bouton des statistiques.

Une fois ce bouton ci-dessus appuyé, plusieurs fonctionnalités seront disponibles.

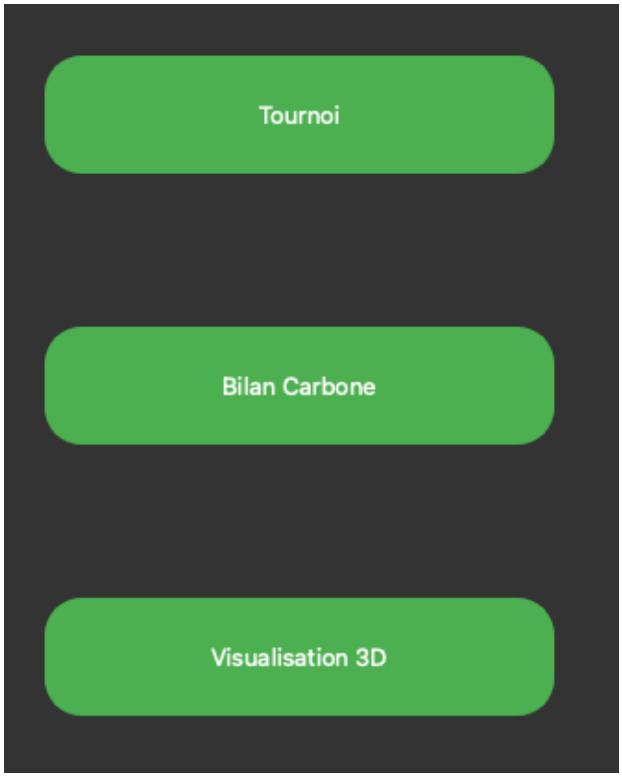


Figure 16 : Options des statistiques.

Une fois le bouton 'Tournoi' appuyé, une nouvelle fenêtre s'ouvrira qui contient les options de tournoi :

Figure 17 : Les options de Tournoi.

D'ici, vous avez le choix entre visualiser l'arbre de tournoi ou inclure un tricheur.

Lorsque vous cliquez sur le bouton 'Arbre Tournoi', l'arbre de ce tournoi s'affiche, révélant ainsi le vainqueur. Alternativement, si vous préférez inclure un tricheur en sélectionnant 'Inclure un tricheur', vous pourrez choisir parmi les candidats celui qui sera désigné comme le tricheur, puis confirmer votre choix en appuyant sur 'Confirmer le Tricheur'. Une fenêtre s'ouvrira pour vous informer du sort du tricheur, que ce soit une victoire ou une défaite.

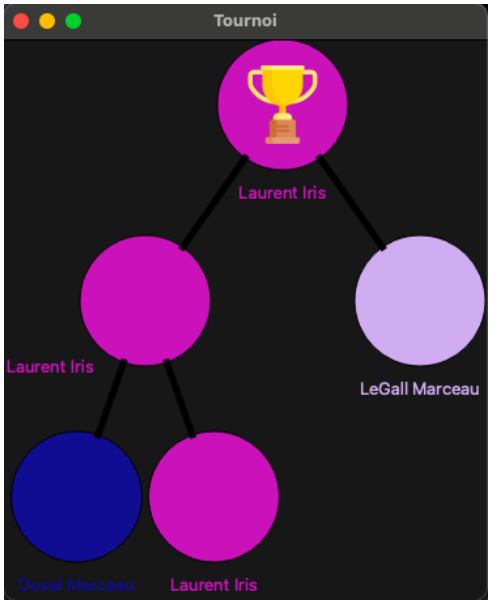


Figure 18 : L'arbre de tournoi.

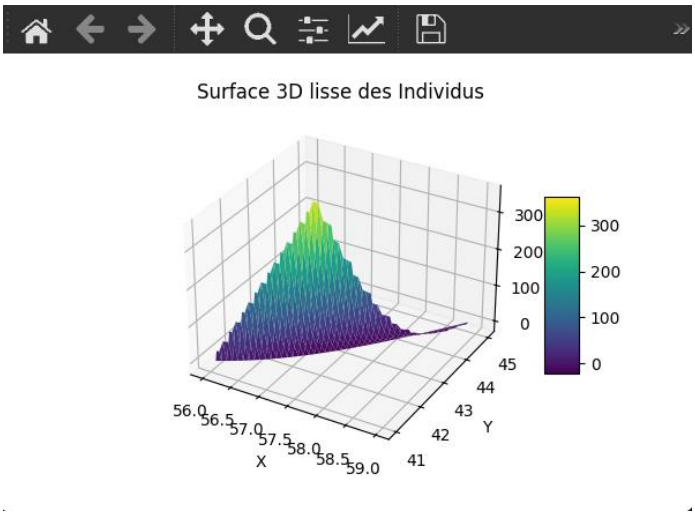


Figure 19 : Surface 3D lisses des Individus

Le bouton 'Visualisation 3D' nous montre le nombre des individus en fonctions de leurs coordonnées x et y (Figure 19). Vous pouvez effectuer un zoom sur une partie précise en cliquant sur  , décaler la surface en cliquant sur  , ajuster les paramètres en cliquant sur  , et enfin sauvegarder le fichier en cliquant sur  .

Le bouton 'Bilan Carbone' (Figure 16) affiche l'émission de carbone depuis le lancement du logiciel.

Figure 20 : Émission avant manipulation

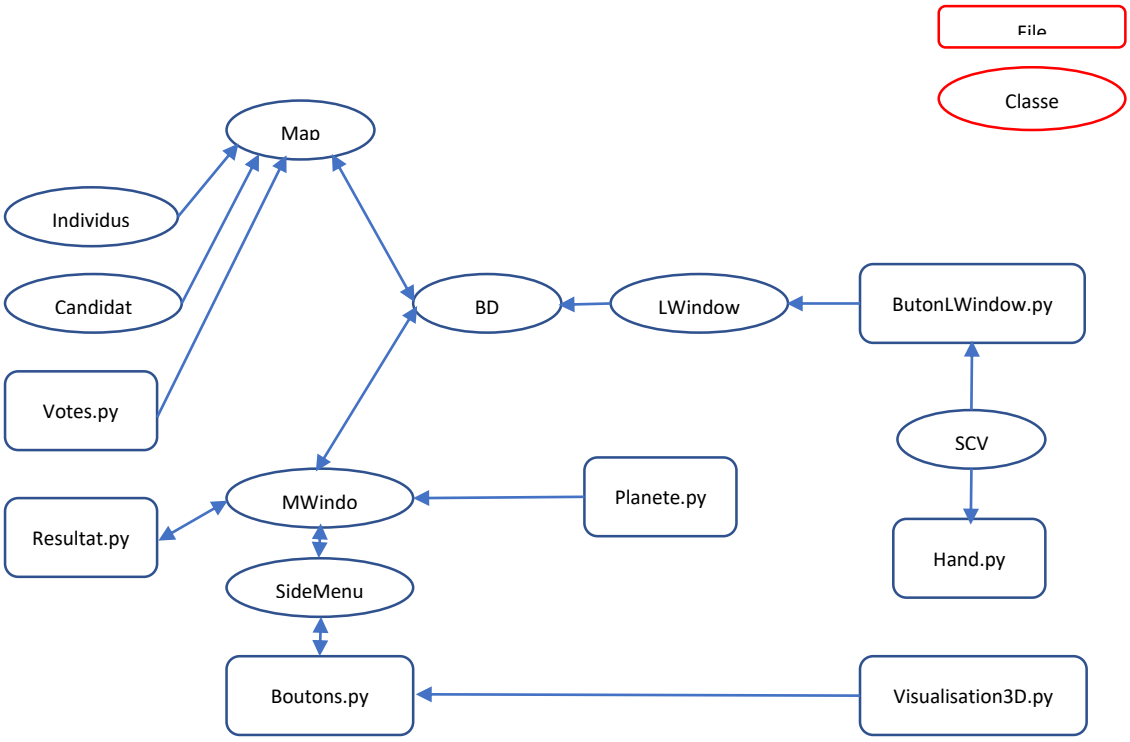
Figure 21 : Émission après manipulation

Une Vision Globale de l'Architecture du Projet

Architecture globale

L'architecture, assez proche de celle qui avait été envisagée lors des premières séances du travail, est centrée sur 4 paquets principaux, chacun ayant des rôles importants pour assurer la bonne implémentation du Compass politique.

- Back
- Front
- Base des donnes
- Tournoi



Structure des modules

Avant de présenter le fonctionnement de l'application, une description des paquetages est nécessaire.

1. Paquetage Back

Les fichiers requis pour réaliser les méthodes d'analyse sur les conditions saisies par l'utilisateur sont inclus dans ce paquetage, ainsi que les classes correspondantes pour leur manipulation et leur traitement. La gestion des différentes étapes du processus de vote, depuis la collecte des préférences des électeurs jusqu'à la détermination des résultats finaux de l'élection, repose sur les classes incluses dans ce paquetage. A part de ces fonctionnalités, le paquetage Backend intègre également un système de logging pour enregistrer les informations pendant l'exécution du programme. Celui-ci est initialisé à l'aide de la fonction basicConfig du module logging, afin d'écrire dans le fichier nommé "app.log" situé au niveau du répertoire d'exécution de l'application. Les messages de logging sont

placés aux endroits spécifiques, par exemple dans le constructeur de la classe Candidat, pour suivre le déroulement du programme.

Classe Candidat :

Cette classe de paquetage représente les candidats proposés dans l'élection, contenant des fonctions d'initialisation et génération.

Classe Individus :

C'est une classe qui définit les votants qui choisissent le vainqueur d'une élection. Ici, on a fait un choix de conception importante : une instance d'Individu contient tous les votants ayant les mêmes coordonnées sur le Compass Politique. Dans la Map, les individus sont stockés sous forme de matrice (List[List[Individus]]) pour améliorer la liaison entre le Back et le Front et par ce choix, on a éliminé le problème de juxtaposition et d'effacement des données précédents.

Classe Map:

Elle est la classe centrale et représente une implémentation d'une carte pour un système électoral. Comme ça, elle est la liaison entre le Back et le Front. Ses fonctionnalités principales sont :

- Initialisation :

-Prend en compte divers paramètres comme la taille de la carte, les listes de candidats et -- d'individus, et les paramètres de génération de population.

- Génération de population :

-Génère une population initiale, soit de manière linéaire soit aléatoire, en fonction de la méthode choisie.
-Permet de générer la population de manière spécifique dans différentes zones avec différentes distributions (triangulaire, uniforme, exponentielle, beta).

- Calcul des votes :

-Calcule le taux de satisfaction d'un candidat.
-Implémente différentes méthodes de vote telles que Copeland, Borda, STV, et Pluralité.
-Gère la méthode de Monte Carlo pour simuler différents types de votes.

- Contraintes et calculs associés :

-Applique une contrainte de délégation de vote pour certains individus.
-Crée des paires de candidats pour comparer leurs performances dans un tournoi.
-Calcule le candidat préféré entre deux options.
-Calcule les victoires d'un candidat dans un tournoi en utilisant la méthode de Copeland.

- Gestion des données :

-Permet de sauvegarder les données de la carte dans un fichier.
-Permet de charger les données à partir d'un fichier.

- Autres fonctionnalités :

-Affichage de la carte.
-Conversion entre liste de population et matrice de population.
-Création de sous-cartes à partir d'une carte carrée donnée.

Fichier votes.py

Le code-source contient des fonctions de calcul et des méthodes pour déterminer le vainqueur dans un système de vote (Pluralité, Borda, Approbation, Approbation par Zone, Single Transferable Vote, Copeland, Simpson). Ici, un choix de conception est que ces fonctions manipulent des listes.

Fichier MonteCarlo.py
Ce fichier fournit un ensemble d'outils pour implémenter la méthode Monte-Carlo, qui simule plusieurs élections sur des sous-zones de la carte principale.
De plus, "MonteCarlo.py" import le module Python "multiprocessing" pour accélérer le processus de la simulation.
Malheureusement, on ne peut pas utiliser cette fonctionnalité à cause d'un problème de dépendance entre mediapipe et openCV(cv2).

Fichier Intelligent/Hand.py
Ce code est compris des classes *FiltreKalmanSimple, Doigt*, *Hand*, *SLH* et *SCV.* Ainsi, il permet de détecter et suivre les mouvements de la main en utilisant OpenCV et MediaPipe et en lissant la distance entre le pouce et les autres doigts.

Fichier Intelligent/SLCV.py
Ce fichier donne des fonctionnalités de traitement vocal de commandes, en utilisant la reconnaissance vocale via l'API Google et la bibliothèque fuzzywuzzy pour la comparaison de chaines. Donc, il permet de reconnaitre les commandes vocales prononces par l'utilisateur et de les associer à des actions définies dans le code.

Fichier Visualisation3D.py
Il comprend des fonctions permettant de visualiser des données en 3D, en utilisant une interpolation polynomiale, et la classe VisualisationThread (QThread) qui gère l'extraction des données pour ne pas bloquer l'interface utilisateur pendant les calculs.

2. Paquetage BaseDonnee

Ce paquetage contient la *Classe Basedonnee* qui interagit avec l'interface utilisateur et gère les opérations de manipulation d'une base des données associe à une carte virtuelle ~ Compass Politique.
Elle a des fonctionnalités très importants comme le rafraîchissement, l'ajout d'un candidat, le sauvegarde et le rechargement.

3. Paquetage Front

Le paquetage Front offre une interface utilisateur interactive pour interagir avec les fonctionnalités de l'application. Elle a été développée avec PySide, qui est une bibliothèque Python qui permet de créer des programmes graphiques en utilisant le framework Qt.

Classe MainWindow
Elle comprend une interface utilisateur intuitive avec des widgets interactifs, un menu latéral et une liste des résultats.
Nous avons intégré une fonctionnalité pour afficher une carte, que ce soit une représentation classique ou une représentation plus intéressante : planétaire, 3D.

Classe LWindow

Elle est la classe centrale de ce paquetage, car LWindow représente la fenêtre principale de l'application, en coordonnant les interactions entre les widgets et les divers composants. Par exemple, elle permet à l'utilisateur de manipuler des différents paramètres pour générer des données personnalisées.

Sous-paquetage Widgets

- *Resultat.py*

Ce fichier contient 2 classes *(BarreScore, ListeResultat)* qui fournissent des widgets pour afficher les scores et les informations des candidats dans une liste, ainsi que des méthodes pour mettre à jour cette liste de manière efficace.

- *Classe SideMenu*

Cette classe comprend des méthodes pour ajouter des boutons de barre latérale, basculer les barres latérales et ouvrir les options de tournoi.

- *Boutons.py*

Ce code offre toutes les fonctionnalités que nous avons programmé dans le Back (entrée de Candidat, Génération Aléatoire de Candidat, Méthode de Vote, Simulation Monte-Carlo, Visualisation 3D, Arbre de Tournoi)

Fichier Planete.py
Ce fichier utilise OpenGL Pour afficher une planète en rotation et permet à utilisateur d'ajouter des points sur sa surface en faisant glisser la souris, alors que nous avons une mise à jour dynamique.

4. Paquetage Tournoi

Classe TreeEllipse – représente un nœud dans une structure d'arbre graphique visualisée sous la forme d'une ellipse.
Classe TreeView - les ellipses et leurs connexions sont organisées pour former un arbre. Aussi, Il calcule la hauteur de l'arbre nécessaire pour un arbre parfaitement équilibré avec le nombre donné de nœuds
Classe Node – représente un nœud dans la structure d'un arbre binaire
Classe Tournoi - gère le déroulement d'un tournoi dans l'arbre de décision, en simulant des matchs entre les candidats

Rapport de tests :

L'objectif de ce rapport est de fournir une évaluation détaillée et concise de la performance de notre logiciel. Nous visons à présenter les différentes étapes et méthodes utilisées pour tester le logiciel, ainsi que les résultats obtenus à partir de ces tests. L'accent sera mis sur l'identification des calculs limites de certains paramètres clés, comme le nombre de candidats et le nombre de votants, et sur la manière dont ces limites ont été calculées. En détaillant cela nous espérons offrir une compréhension claire des performances du logiciel.

Nos fichiers de jeux de tests sont dans le fichier 'Test'.

Cette partie sera divisée en plusieurs sous-parties, contenant notamment :

1. Méthodologie
2. Paramètres de tests
3. Résultats
4. Conclusion

Méthodologie :

Dans le cadre du développement de notre projet, nous avons utilisé une variété de types de tests pour garantir la qualité et la fiabilité de notre code. Ces tests comprennent des tests unitaires, des tests d'intégration, des tests fonctionnels, des tests de validation et des tests de performance.

Les tests unitaires ont été largement employés pour évaluer le comportement des différentes parties de notre code de manière isolée. Chaque fichier contenant des fonctions ou des méthodes a été accompagné d'une suite de tests unitaires visant à valider son fonctionnement dans des conditions spécifiques. Par exemple, dans le fichier de la classe Map 'test_map', des tests comme test_distance, test_ajoute_candidat, et test_liste_to_matrice sont des exemples de tests unitaires qui vérifient le comportement des méthodes individuelles de la classe. Dans le fichier de test de vote 'test_votes', chaque méthode de votes a été testé individuellement pour vérifier si elle renvoie le résultat attendu.

Pour garantir la qualité et la robustesse de notre projet, nous avons également effectué des tests de validation pour vérifier que les résultats produits par notre code sont corrects et conformes aux attentes. Ces tests incluent des assertions pour vérifier les types de retour et les valeurs renvoyées par les fonctions. Par exemple, dans 'test_utilitaire' on vérifie que les valeurs renvoyées sont du bon type et que les structures de données renvoyées sont correctement formatées. Des blocs try et except sont utilisés dans 'test_votes' pour capturer les exceptions qui pourraient être levées lors de l'exécution des méthodes de vote. Cela permet de détecter et de gérer les erreurs éventuelles.

En plus, nous avons également réalisé des tests d'intégration pour évaluer la manière dont les différentes parties de notre application interagissent les unes avec les autres. Par exemple, des tests comme 'test_listes_listes_votes' dans la classe Map examinent comment plusieurs méthodes interagissent ensemble telles que la création des candidats, la

génération de la population, la création de l'instance de la classe Map, et l'appel des différentes méthodes de vote.

Enfin, des tests de performance ont été réalisés pour évaluer la réactivité et l'efficacité de notre application dans des conditions réelles d'utilisation. Ces tests ont permis d'identifier et de corriger les problèmes de performance (testPL et testSideBarre).

En combinant ces différents types de tests, nous avons pu assurer une couverture complète de notre application et garantir sa qualité, sa fiabilité et sa conformité aux spécifications tout au long du processus de développement.

Paramètres de test :

Dans le cadre des paramètres de test, il est important de noter que pour les candidats, il n'y a pas de limites spécifiées. Cette absence de limite vise à garantir la flexibilité du système pour s'adapter à un nombre variable de candidats dans différentes situations d'utilisation. Cependant, en ce qui concerne la génération d'individus, dès que le nombre dépasse 1 million, le processus commence à prendre significativement plus du temps, prenant un peu plus qu'une minute.
Les fonctions de vote sont également contraintes à un maximum de 10 000 opérations, permettant ainsi de manipuler un grand nombre d'individus sans problème, même jusqu'à 1 milliard, grâce à la complexité en temps constant (O(1)) de la fonction len() en Python.
Pour générer un tournoi, des centaines de candidats peuvent être utilisés, et pour déterminer un gagnant dans un arbre de tournoi, au maximum 10 candidats sont en compétition.

Discussion des résultats :

Après avoir examiné les résultats de nos tests, nous pouvons affirmer que la plupart des fonctionnalités du logiciel se comportent conformément aux spécifications et produisent les résultats attendus. Par exemple, les tests de création d'individu ont confirmé que les instances de la classe Individus sont correctement créés avec les attributs spécifiés tels que le nom, les coordonnées et la liste des électeurs. De même, les tests de génération de liste de vote ont démontré que les électeurs choisissent les candidats dans un ordre conforme à leurs préférences, comme spécifié dans le système de vote.

Conclusion :

Dans cette étude, nous avons examiné en profondeur le logiciel de vote, en nous concentrant sur son fonctionnement, sa fiabilité et ses performances dans différents contextes. À travers une série de tests rigoureux, nous avons évalué sa capacité à gérer un nombre varié de candidats et de votants, ainsi que sa résistance face à des conditions diverses.
Les résultats obtenus démontrent que le logiciel répond globalement aux attentes fixées. Il est capable de traiter efficacement un nombre conséquent de candidats et de votants, offrant ainsi une solution viable pour la gestion des élections.

Rapport Carbone

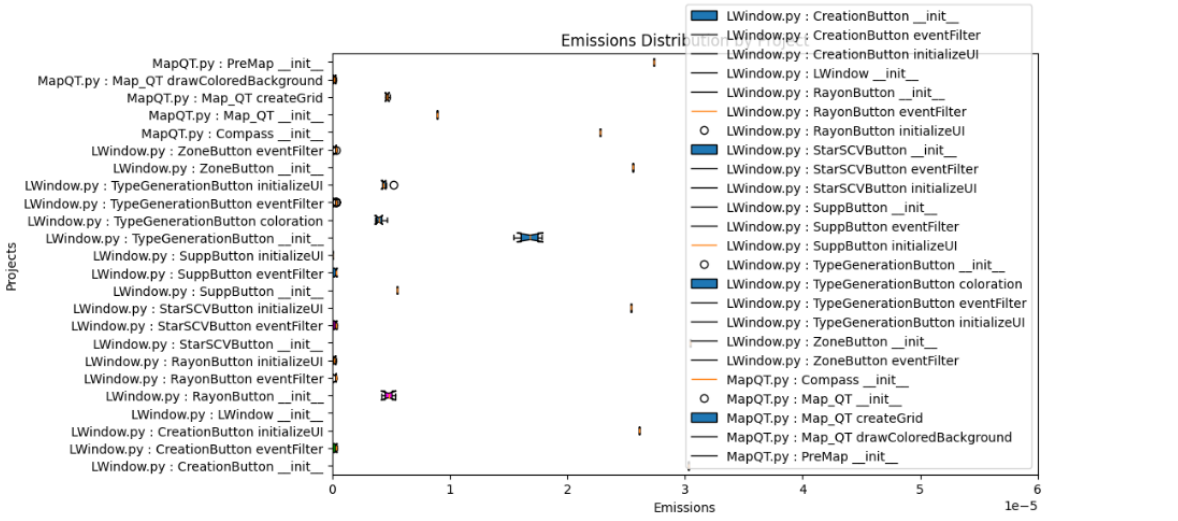
Dans le cadre de ce projet, une évaluation de l'impact carbone est réalisée en lien avec l'exécution du code et en se basant sur une estimation du système de production électrique du pays où le code est exécuté. Cette évaluation repose sur l'utilisation de l'outil CodeCarbon, qui permet de localiser l'exécution du code et d'estimer son empreinte carbone en fonction du mix énergétique du pays.

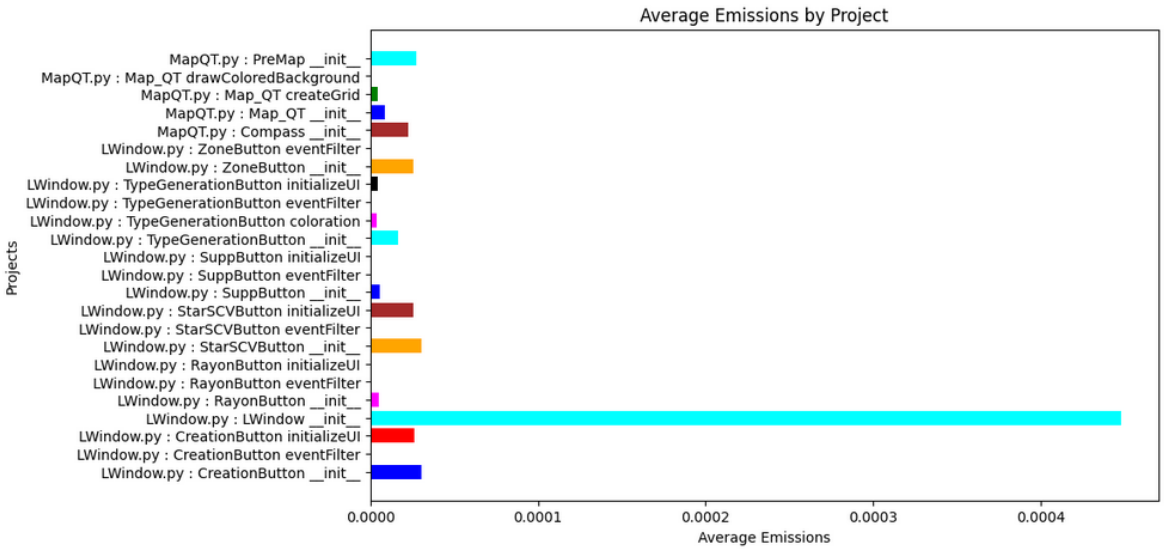
En effet, nous avons ajouté :

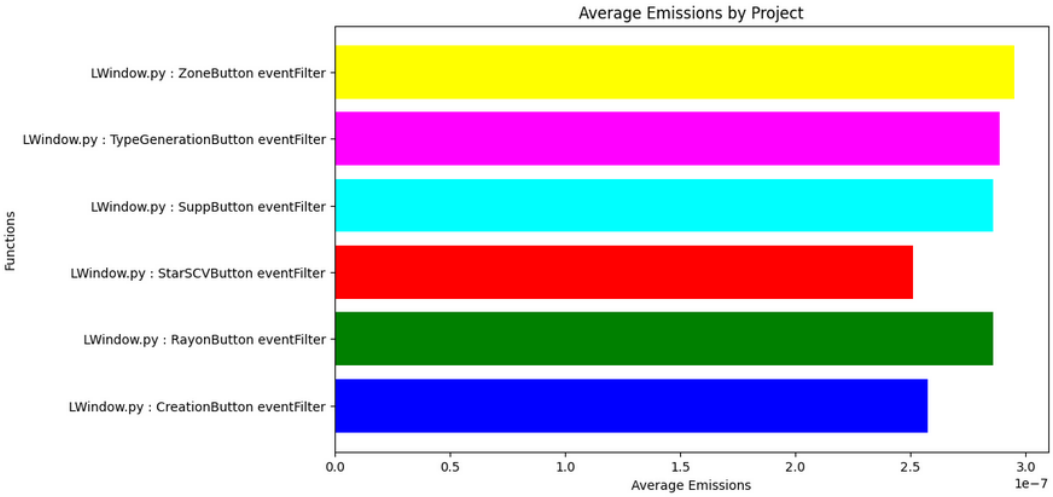
@track_emissions (project_name=<nom_de_la_fonction>)

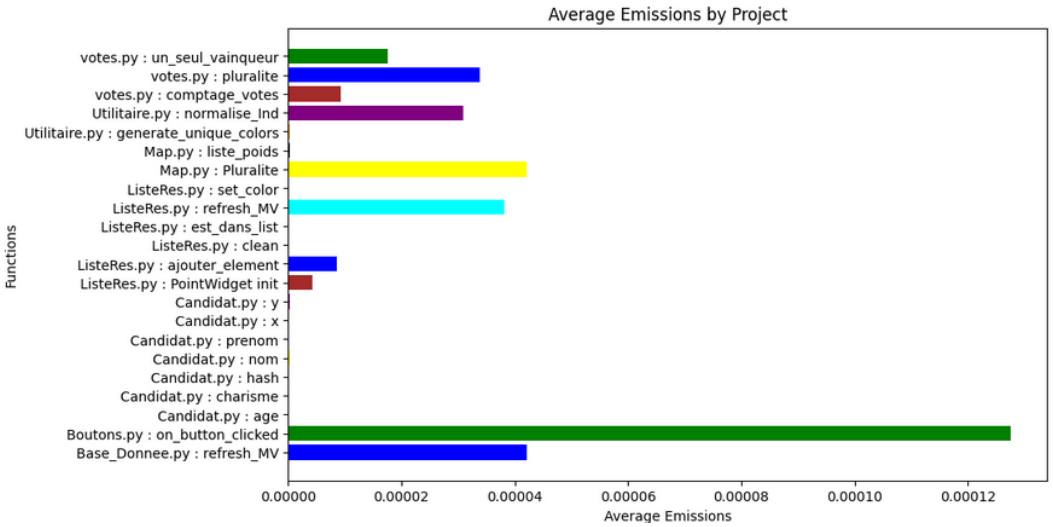
devant toutes les fonctions et nous avons lancé plusieurs fois notre logiciel pour récupérer les mesures dans un fichier emissions.csv. Elles ont été transférées dans plusieurs fichiers csv .

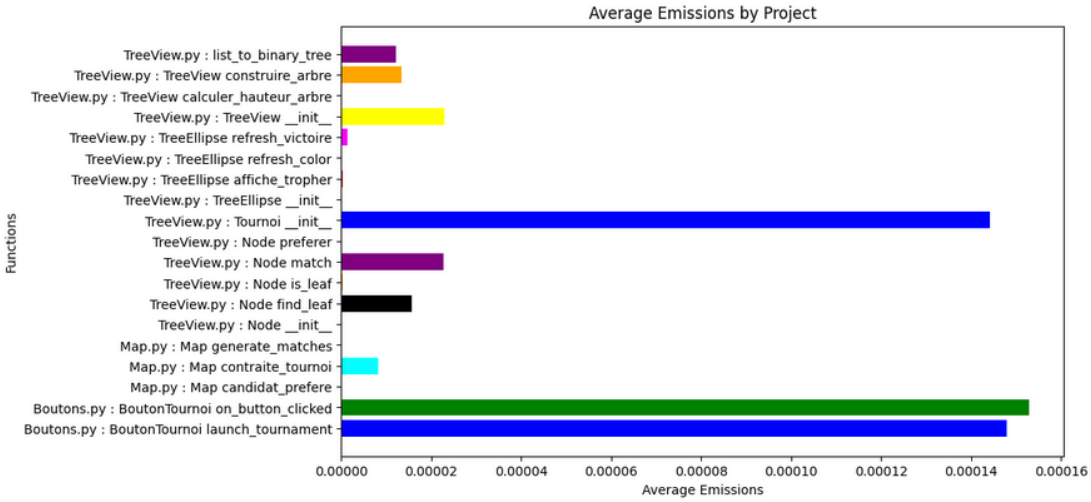
De plus, l'application a un fichier avec des fonctions qui calculent l'émission moyen de chacun des fonctionnalités de notre logiciel. Ces moyennes sont récupérées dans des variables globales. En bas, j'ajoute 2 types différents de visualisation des données, pour remarquer quelles fonctions sont les plus 'couteux'.

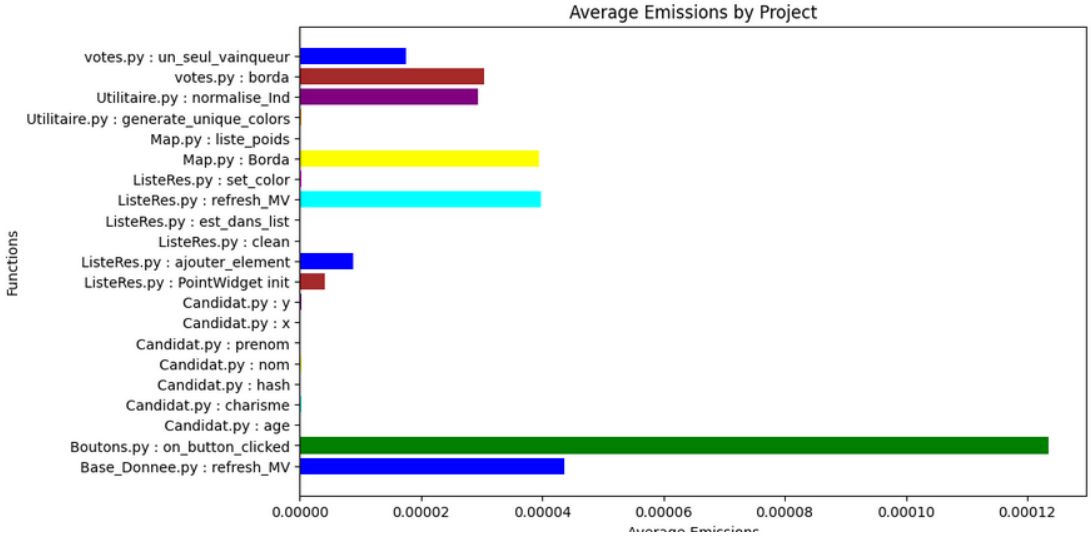


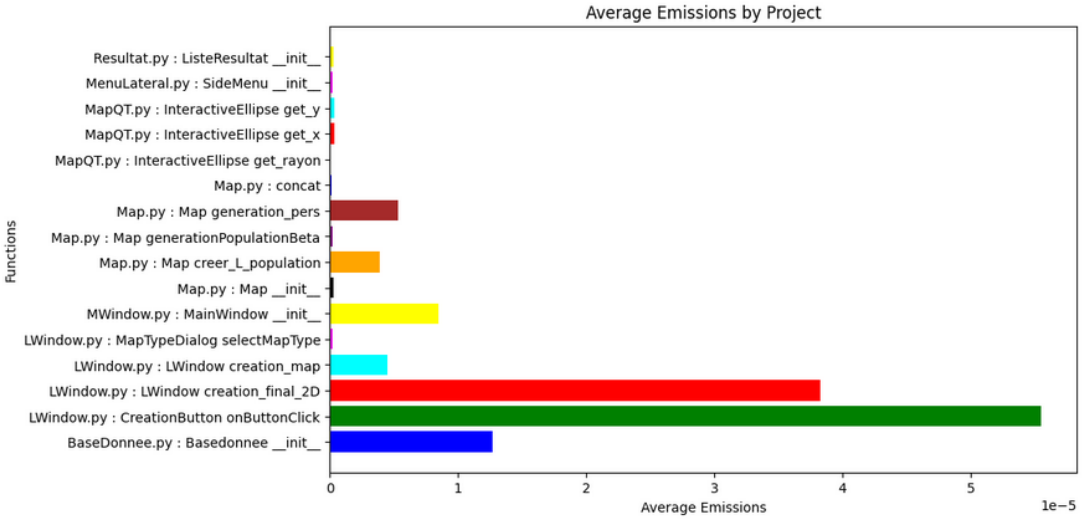



Average Emissions by Project
LWindow.py : ZoneButton eventFilter
LWindow.py : TypeGenerationButton eventFilter
LWindow.py : SuppButton eventFilter
LWindow.py : StarSCVButton eventFilter
LWindow.py : RayonButton eventFilter
LWindow.py : CreationButton eventFilter
Functions
0.0
0.5
1.0
1.5
2.0
2.5
3.0
1e−7
Average Emissions


Average Emissions by Project
votes.py : un_seul_vainqueur
votes.py : pluralite
votes.py : comptage_votes
Utilitaire.py : normalise_Ind
Utilitaire.py : generate_unique_colors
Map.py : liste_poids
Map.py : Pluralite
ListeRes.py : set_color
ListeRes.py : refresh_MV
ListeRes.py : est_dans_list
ListeRes.py : clean
ListeRes.py : ajouter_element
ListeRes.py : PointWidget init
Candidat.py : y
Candidat.py : x
Candidat.py : prenom
Candidat.py : nom
Candidat.py : hash
Candidat.py : charisme
Candidat.py : age
Boutons.py : on_button_clicked
Base_Donnee.py : refresh_MV
Functions
0.00000
0.00002
0.00004
0.00006
0.00008
0.00010
0.00012
Average Emissions


Average Emissions by Project
TreeView.py : list_to_binary_tree
TreeView.py : TreeView construire_arbre
TreeView.py : TreeView calculer_hauteur_arbre
TreeView.py : TreeView __init__
TreeView.py : TreeEllipse refresh_victoire
TreeView.py : TreeEllipse refresh_color
TreeView.py : TreeEllipse affiche_tropher
TreeView.py : TreeEllipse __init__
TreeView.py : Tournoi __init__
TreeView.py : Node preferer
TreeView.py : Node match
TreeView.py : Node is_leaf
TreeView.py : Node find_leaf
TreeView.py : Node __init__
Map.py : Map generate_matches
Map.py : Map contraite_tournoi
Map.py : Map candidat_prefere
Boutons.py : BoutonTournoi on_button_clicked
Boutons.py : BoutonTournoi launch_tournament
Functions
0.00000
0.00002
0.00004
0.00006
0.00008
0.00010
0.00012
0.00014
0.00016
Average Emissions


Average Emissions by Project
votes.py : un_seul_vainqueur
votes.py : borda
Utilitaire.py : normalise_Ind
Utilitaire.py : generate_unique_colors
Map.py : liste_poids
Map.py : Borda
ListeRes.py : set_color
ListeRes.py : refresh_MV
ListeRes.py : est_dans_list
ListeRes.py : clean
ListeRes.py : ajouter_element
ListeRes.py : PointWidget init
Candidat.py : y
Candidat.py : x
Candidat.py : prenom
Candidat.py : nom
Candidat.py : hash
Candidat.py : charisme
Candidat.py : age
Boutons.py : on_button_clicked
Base_Donnee.py : refresh_MV
Functions
0.00000
0.00002
0.00004
0.00006
0.00008
0.00010
0.00012


Average Emissions by Project
Resultat.py : ListeResultat __init__
MenuLateral.py : SideMenu __init__
MapQT.py : InteractiveEllipse get_y
MapQT.py : InteractiveEllipse get_x
MapQT.py : InteractiveEllipse get_rayon
Map.py : concat
Map.py : Map generation_pers
Map.py : Map generationPopulationBeta
Map.py : Map creer_L_population
Map.py : Map __init__
MWindow.py : MainWindow __init__
LWindow.py : MapTypeDialog selectMapType
LWindow.py : LWindow creation_map
LWindow.py : LWindow creation_final_2D
LWindow.py : CreationButton onButtonClick
BaseDonnee.py : Basedonnee __init__
Functions
0
1
2
3
4
5
1e−5
Average Emissions